Security Analyse mithilfe von Codex Security

„Kann ja nicht so schwer sein…“

Mika Schmidt

byte5 Meetup

Six Eight CONSULTING

#Werbung

Über mich

Mika Schmidt

- Seit 2021 in der IT-Security
- Fokus: Security für E-Commerce
- 2025 „Six Eight Consulting“ gegründet



Six Eight CONSULTING

„Ich habe eine App gevibecodet. Ich brauche Security.“



Six Eight CONSULTING

Sein Ziel: Wissen, dass seine App sicher ist

Mein Ziel: Herausfinden, wie ich diese Security-Reviews mit Codex durchführen kann

Die Ergebnisse

20

Potentielle Sicherheitslücken

9

verifizierte Befunde

7 Live-Probes + 2 Konfigurationslücken

1

verifizierter Angriffspfad

Von „kein Zugriff“ zu Projekt- und Kundendaten

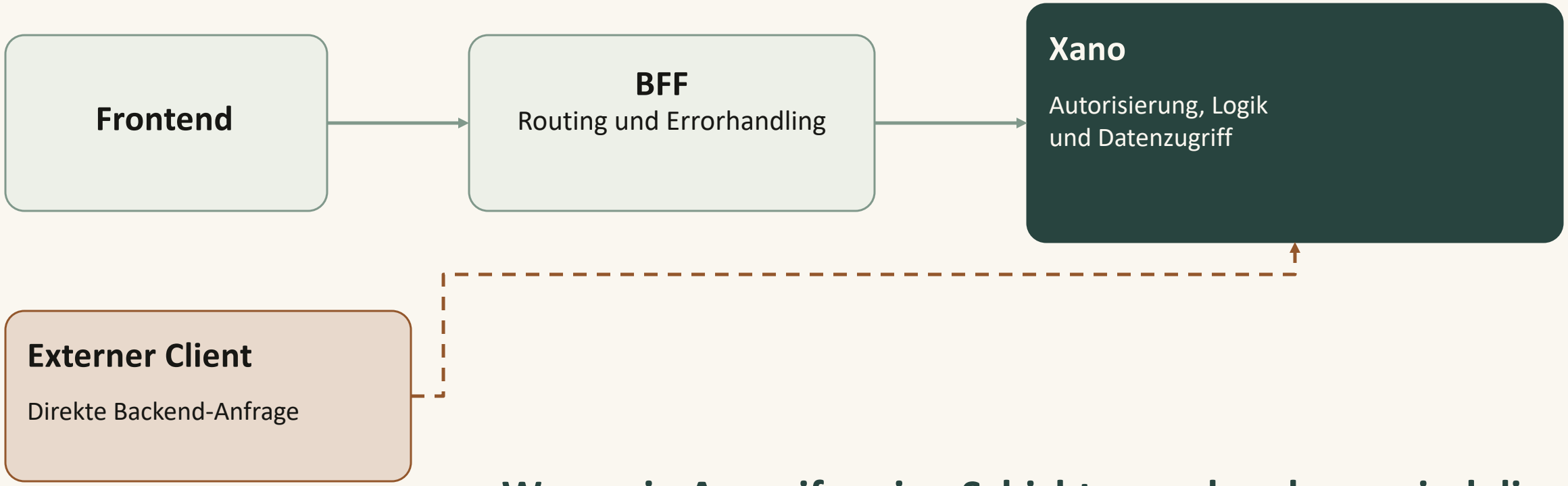
Six Eight CONSULTING

Mein erster Prompt

„Finde Sicherheitslücken“

Das hat nicht funktioniert.

Auf was kommt es wirklich an?



Wenn ein Angreifer eine Schicht umgehen kann, sind die Sicherheitsmechanismen dort egal.

Entdeckung und Validierung

ENTDECKUNG



Wo könnte eine Kontrolle fehlen?

VALIDIERUNG

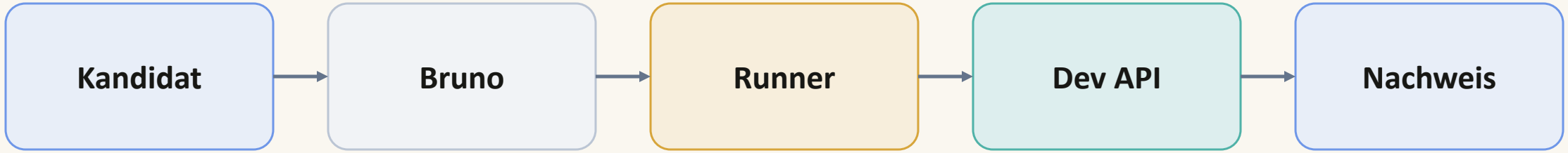


Wie lässt sich das prüfen?

Ein Agent sammelt Kandidaten, die anderen prüfen Erreichbarkeit, nachgelagerte Kontrollen und Gegenbeweise.

Six Eight CONSULTING

Mein Verifikations-Framework



Bruno

Reproduzierbare Testfälle

Python-Runner

Dry Run, Ausführungs-Gates
und isolierte Schreibtests

Nachweis

Geschwärzte Requests,
Responses
und beobachteter Zustand

Six Eight
CONSULTING

Fragen, man stellen sollte

Scope

Welche produktiven Angriffsflächen sind relevant?

Angreifer

Welche Rollen und Voraussetzungen?

Domainwissen

Wer darf welches Objekt lesen oder ändern?

Nachweis

Welche Beobachtung bestätigt oder widerlegt die Behauptung?

Das Codex-Security-Plugin für große Codebasen nutzen

6

Sub-Agents pro Run

Klar abgegrenzte Anweisungen für jeden Agent

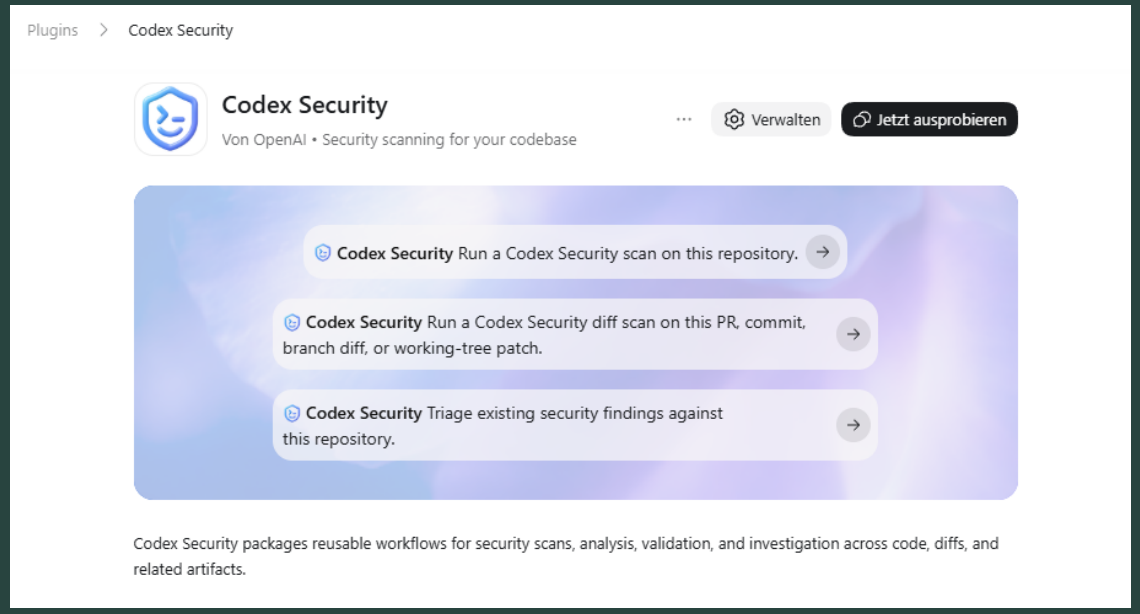
Gemeinsame Rollen, Vertrauensgrenzen und Nachweisschema

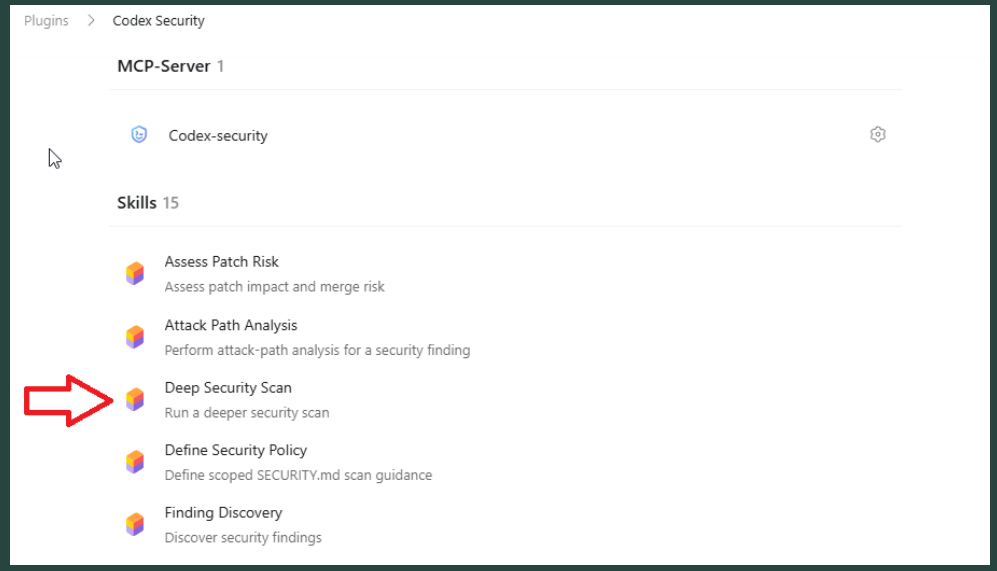
Gemeinsame Dokumentation für Kandidaten, Duplikate und Abdeckungslücken

https://chatgpt.com/plugins/codex-security

Six Eight CONSULTING

Das Codex-Security-Plugin für große Codebasen nutzen





Six Eight CONSULTING

Das Codex-Security-Plugin für große Codebasen nutzen

Führe mit dem Codex-Security-Plugin einen **Deep Security Scan ausschließlich für das Xano-Backend dieses Projekts** durch.

Scope

- Untersuche alle verfügbaren Xano-Artefakte, insbesondere API-Endpunkte, Function Stacks, Datenbankzugriffe, Authentifizierung, Autorisierung, Rollen- und Rechteprüfungen, Inputs, Add-ons, Background Tasks, externe API-Aufrufe, Secrets sowie Datei-Uploads.
- Berücksichtige die Beziehungen zwischen Endpunkten, Funktionen und Datenbanktabellen, um auch mehrstufige Angriffspfade zu erkennen.

Explizit außerhalb des Scopes

- Das gesamte Astro-Frontend und dessen Komponenten
- Clientseitiger JavaScript-, HTML- und CSS-Code
- SEO, UI, Accessibility und allgemeine Codequalität
- Frontend-Abhängigkeiten und rein clientseitige Schwachstellen
- Deployment-Konfigurationen des Frontends

Verwende Frontend-Code höchstens dann als Kontext, wenn er notwendig ist, um einen konkreten Aufruf des Xano-Backends zu verstehen. Erstelle daraus keine eigenständigen Frontend-Findings und leite die Sicherheit eines Xano-Endpunkts nicht aus dem Verhalten des Frontends ab.

Prüfschwerpunkte

1. Fehlende oder umgehbare Authentifizierung
2. Broken Access Control, IDOR/BOLA und fehlende Object-Level Authorization
3. Privilege Escalation und fehlerhafte Rollenprüfungen
4. Unzureichende Eingabevalidierung und Injection-Angriffe
5. Unbeabsichtigte Offenlegung sensibler Daten
6. Unsichere Datei-Uploads
7. SSRF und unsichere externe API-Aufrufe
8. Hardcodierte oder offengelegte Secrets
9. Fehlende Rate Limits und Missbrauchsmöglichkeiten
10. Unsichere Geschäftslogik und mehrstufige Angriffspfade

Führe einen **vollständigen Deep Scan mit mehreren unabhängigen Scan-Durchläufen** durch. Validiere mögliche Findings anhand der tatsächlich vorhandenen Xano-Konfiguration und reduziere False Positives.

Dokumentiere pro bestätigtem Finding:

- Titel und Schweregrad
- betroffener Xano-Endpunkt beziehungsweise Function Stack
- konkrete technische Ursache
- nachvollziehbarer Angriffspfad
- Voraussetzungen und mögliche Auswirkungen
- Belege aus den untersuchten Artefakten
- konkrete Empfehlung zur Behebung
- verbleibende Unsicherheiten

Trenne bestätigte Schwachstellen klar von Vermutungen und nicht überprüfbaren Risiken. Wenn die vorhandenen Xano-Artefakte für einen vollständigen Scan nicht ausreichen, liste zuerst präzise auf, welche Exporte oder Informationen fehlen. Verändere während des Scans keinen Code und keine Xano-Konfiguration.

Ok, dann legen wir los…..



Six Eight CONSULTING

Kontext- und Token-Limits



Wiederanlauffähige Checkpoints umsetzen

Getestete Revisionen und Umgebungen verwenden

Finding-IDs, Nachweise und Gegenbeweise, verbleibende Beweislücke und nächster Test

Review-Status außerhalb des Chats halten.
Quelle bei Neustart einer Sitzung erneut öffnen.

Nächster Versuch mit allem Gelernten…



Six Eight CONSULTING

Die Ergebnisse

20

Potentielle Sicherheitslücken

9

verifizierte Befunde

7 Live-Probes + 2 Konfigurationslücken

1

verifizierter Angriffspfad

Von „kein Zugriff“ zu Projekt- und Kundendaten

Six Eight CONSULTING

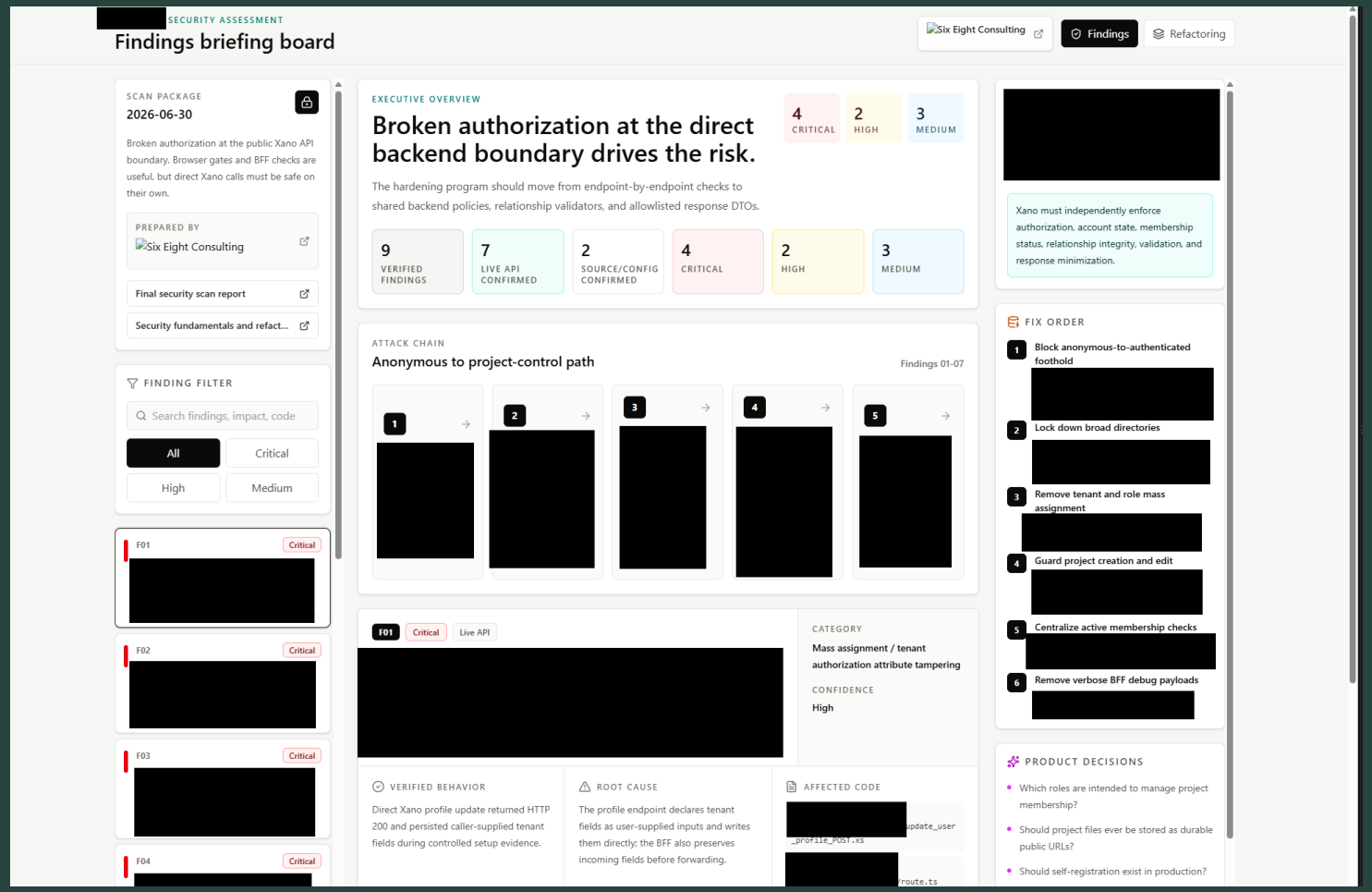
Wie präsentiert man die Ergebnisse?

Einfach für den Kunden, detailliert für die Agents.



Six Eight CONSULTING

Kunden-Briefing: Übersichtliche Visualisierung



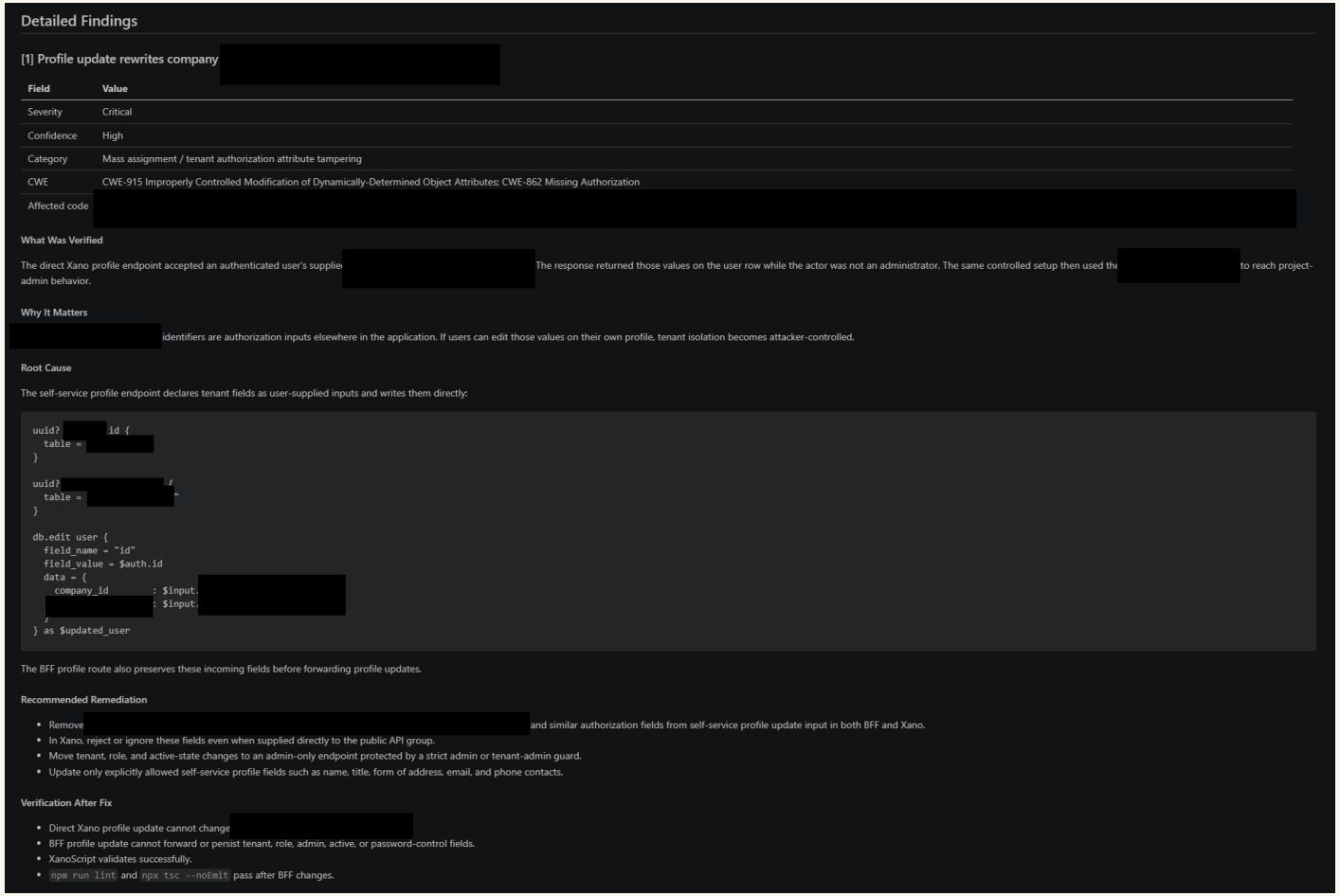
FÜR ENTSCHEIDUNGEN

Risiko
Angriffspfad
Reihenfolge der Fixes

Klare Sprache mit eindeutigen Prioritäten.

Six Eight CONSULTING

Agent-Übergabe: Markdown pro Finding



Detailed Findings

[1] Profile update rewrites company [redacted]

Field	Value
Severity	Critical
Confidence	High
Category	Mass assignment / tenant authorization attribute tampering
CWE	CWE-915 Improperly Controlled Modification of Dynamically-Determined Object Attributes; CWE-862 Missing Authorization
Affected code	[redacted]

What Was Verified

The direct Xano profile endpoint accepted an authenticated user's supplie[redacted] The response returned those values on the user row while the actor was not an administrator. The same controlled setup then used th[redacted] to reach project-admin behavior.

Why It Matters

[redacted] identifiers are authorization inputs elsewhere in the application. If users can edit those values on their own profile, tenant isolation becomes attacker-controlled.

Root Cause

The self-service profile endpoint declares tenant fields as user-supplied inputs and writes them directly:

```
uuid? [redacted] id {
  table = [redacted]
}

uuid? [redacted] {
  table = [redacted]
}

db.edit user {
  field_name = "id"
  field_value = $auth.id
  data = {
    company_id        : $input.[redacted]
    [redacted]        : $input.[redacted]
  }
} as $updated_user
```

The BFF profile route also preserves these incoming fields before forwarding profile updates.

Recommended Remediation

- Remove [redacted] and similar authorization fields from self-service profile update input in both BFF and Xano.
- In Xano, reject or ignore these fields even when supplied directly to the public API group.
- Move tenant, role, and active-state changes to an admin-only endpoint protected by a strict admin or tenant-admin guard.
- Update only explicitly allowed self-service profile fields such as name, title, form of address, email, and phone contacts.

Verification After Fix

- Direct Xano profile update cannot change [redacted]
- BFF profile update cannot forward or persist tenant, role, admin, active, or password-control fields.
- XanoScript validates successfully.
- `npm run lint` and `npx tsc --noEmit` pass after BFF changes.

FÜR DIE UMSETZUNG

Verifiziertes Verhalten
Grundursache
Betroffener Code
Behebung
Retest

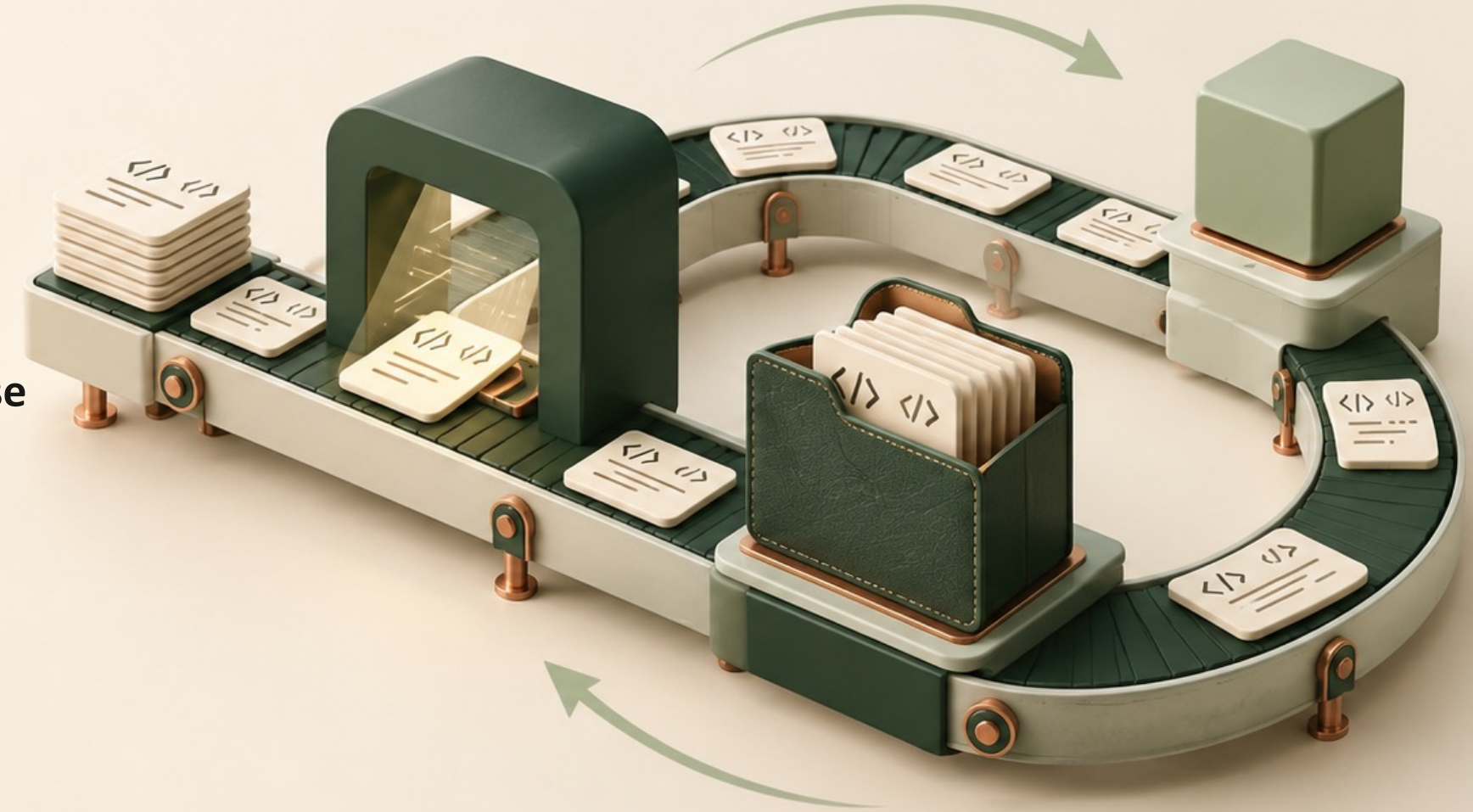
Six Eight CONSULTING

Wie könnte man diese Methode im Unternehmen nutzen?

Kontinuierliche Reviews in CI/CD

AUTOMATISIEREN

Review geänderten Codes
Kandidaten-Diff
Sichere Lese-Probes
JSON- + Markdown-Nachweise

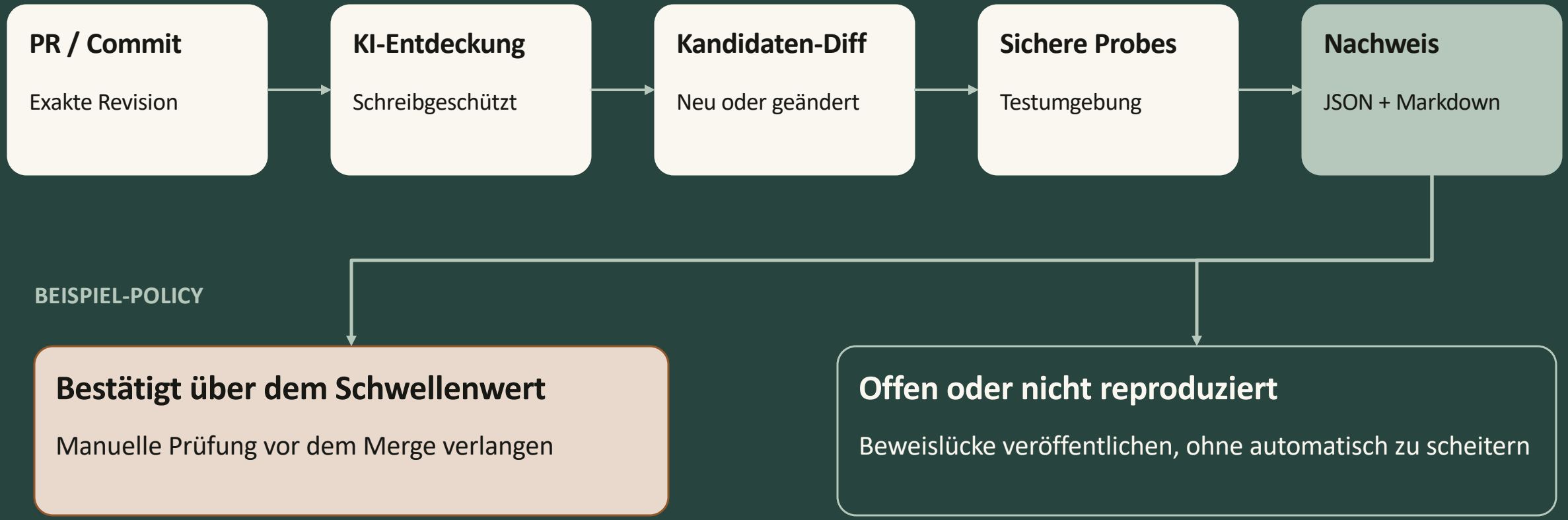


FREIGABE ERFORDERLICH

Verändernde Tests
Produktionszugriff
Schweregrad, Geschäftsziel

Six Eight CONSULTING

Ein Beispiel für ein CI/CD-Security-Gate



Secrets schwärzen und automatisierte Probes nie versehentlich auf Produktion richten.

Six Eight CONSULTING

Wie könnt ihr das privat für eure Projekte nutzen?

1 · Einmal einrichten

Kurze review-brief.md ins Repo: Was ist öffentlich, wer darf was?

2 · Regelmäßig prüfen

Vor jedem Release Codex mit dem Briefing über die Änderungen schicken

3 · Selbst verifizieren

Jeden Kandidaten lokal nachstellen, nie gegen Produktion

4 · Nebenbei absichern

Dependabot und Secret Scanning aktivieren, Findings als Markdown sammeln oder direkt PRs öffnen lassen

Six Eight CONSULTING

Was ich gelernt habe

01 Herausfinden, welche Teile einer App wirklich zählen

02 Entdeckung von Validierung trennen

03 Nachweise außerhalb des Chats halten

04 Artefakte für Übergabe und CI/CD wiederverwenden

„Kann ja nicht so schwer sein…“

Stimmt. Wenn man kurz darüber nachdenkt.

Folgt mir auf LinkedIn!

Fragen?

Six Eight CONSULTING